Demo Starwars 2021

Projet pour démontrer quelques fonctionnalités de Unity :

Table des matières

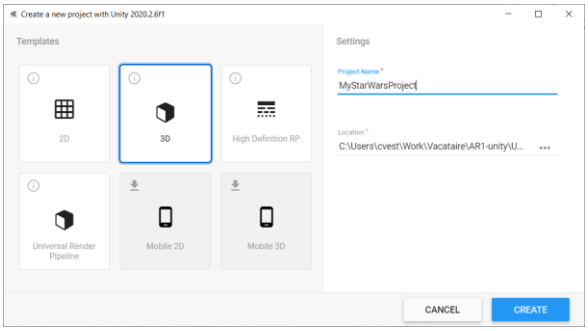
Projet principal pour TP

Voici quelques exemples de fonctionnalités intéressantes pour votre projet , à réaliser ensemble en TP

Récupérer les Assets :

https://github.com/vestri/CoursAR/tree/master/2020/Materiel-ProjetStarWars (il a tout normalement)

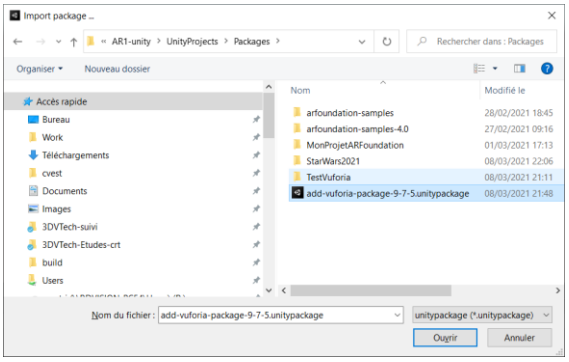
Créer un nouveau projet

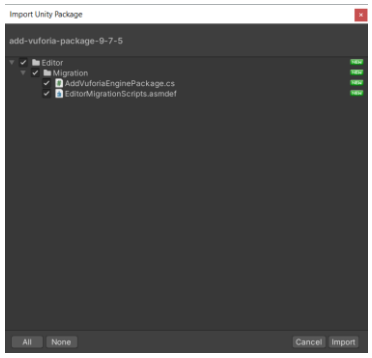


Ajouter le package Vuforia Engine

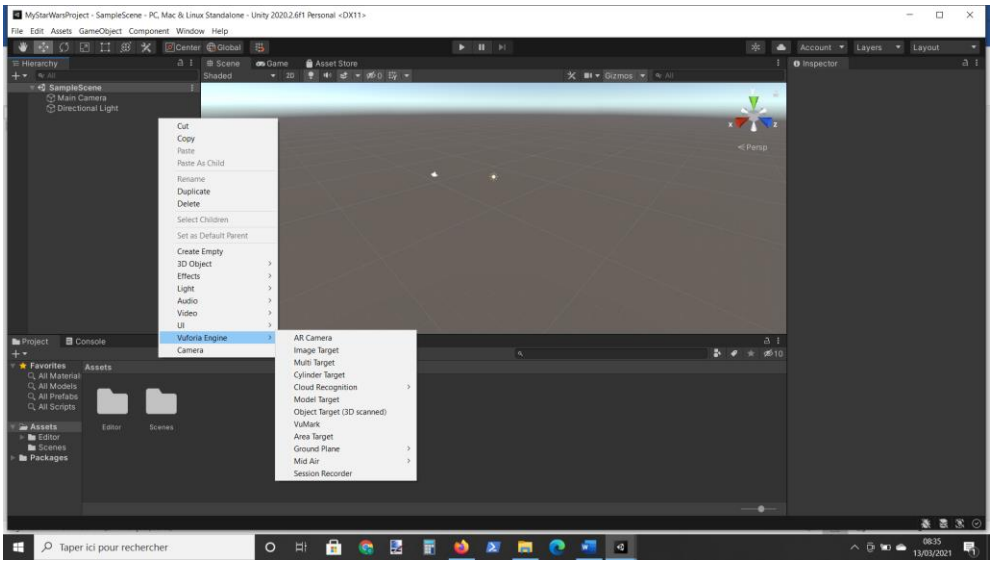
- Il faut d'abord récupérer le dernier package installer sur https://developer.vuforia.com/downloads/sdk
- Il vous faudra certainement un compte Vuforia, créer en un il faudra une licence

- Pour l'installer : Menu Assets/import Package/Custom Package et choisissez le bon unity package : add-vuforia-package-9-7-5.unitypackage (10/03/2021), puis clickez sur import puis update
- Attention l'étape « resolving packages » peut prendre longtemps





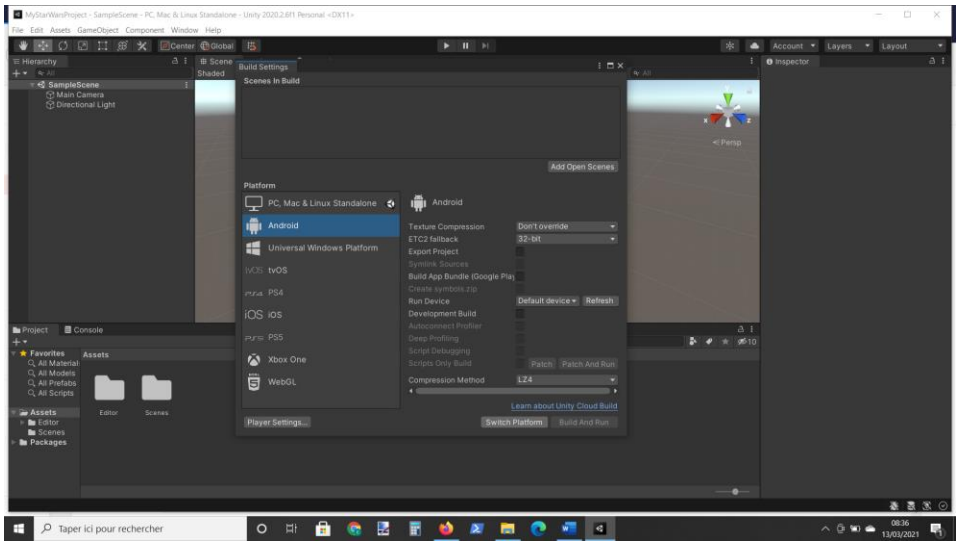
- A la fin il ne devrait pas y avoir d'erreur dans la console et vous devriez avoir le menu VuforiaEngine dans le menu d'ajout d'objets



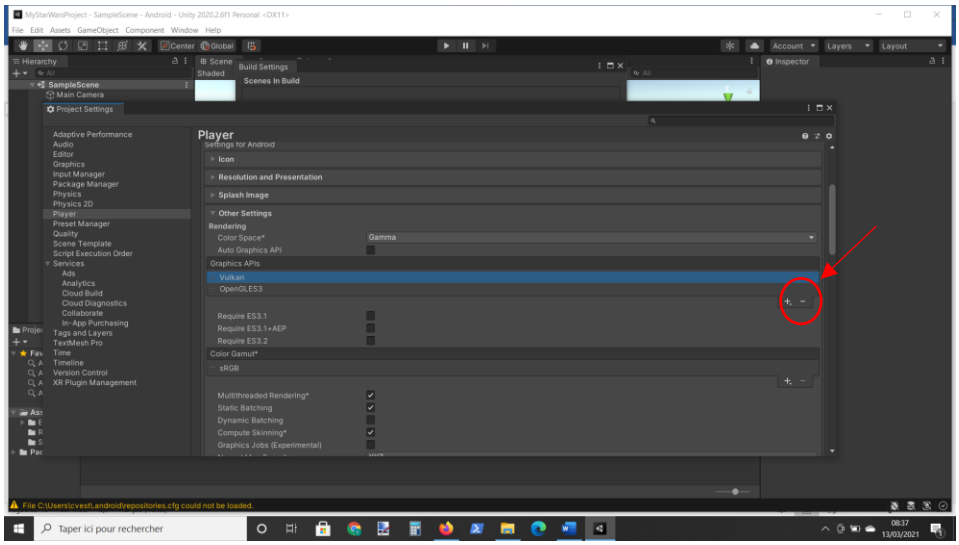
- Si vous avez une/des erreurs(s), quittez et relancez, sinon, réinstallez VuforiaEngine ou créez un nouveau projet et recommencez.

Configuration plateforme

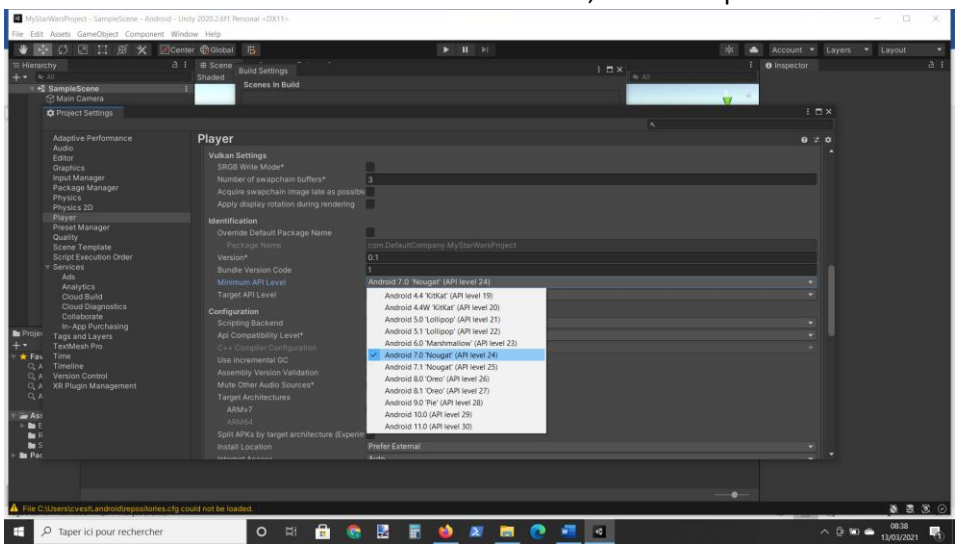
- Ensuite il faut changer la cible d'installation de l'application
- Dans le menu build setting, choisissez Android (ou IOS si Iphone), puis cliquez sur switch platform



- Cliquez sur Player Settings/Other Settings, il faut tout d'abord enlever Vulkan (pour Android) qui pose quelquefois des problèmes d'affichage, sélectionnez Vulcan et cliquez sur –

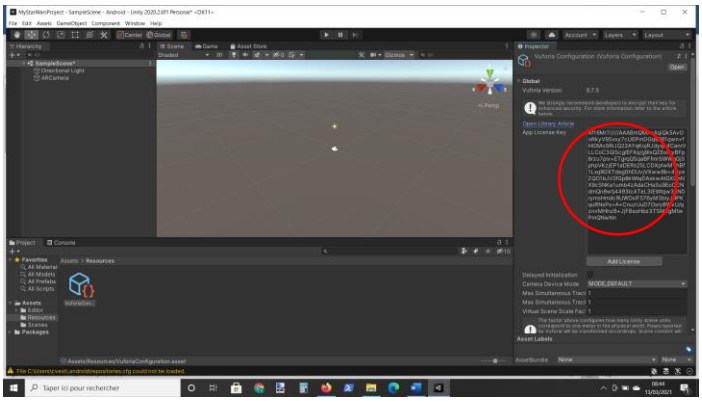


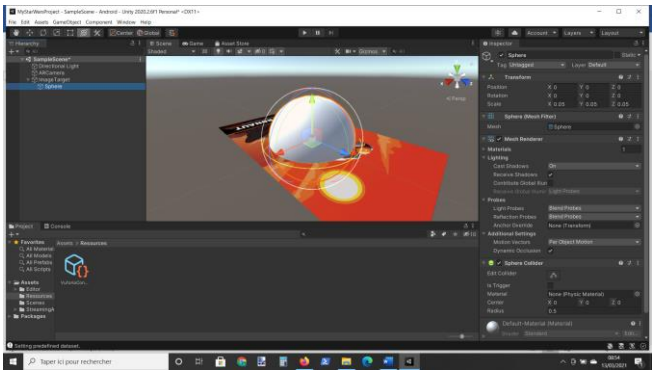
- Choisissez ensuite une API Android assez récente, il semble que la 24 fasse l'affaire

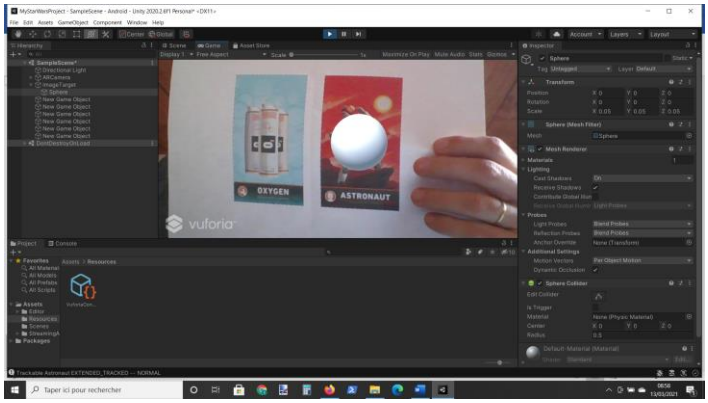


Création d'un scene Vuforia ImageTarget

- Enlever la MainCamera qui n'est plus utile
- Pour tester Image Target il vous faudra ajouter des objets de VuforiaEngine
    - Ajouter une CameraAR
        - Ajouter la licence, dans inspector de l'AR camera, cliquez sur « Open Vuforia Engine Configuration »
        - Cliquez sur Add Licence
        - Récupérez une licence sur le site de Vuforia et ajoutez là



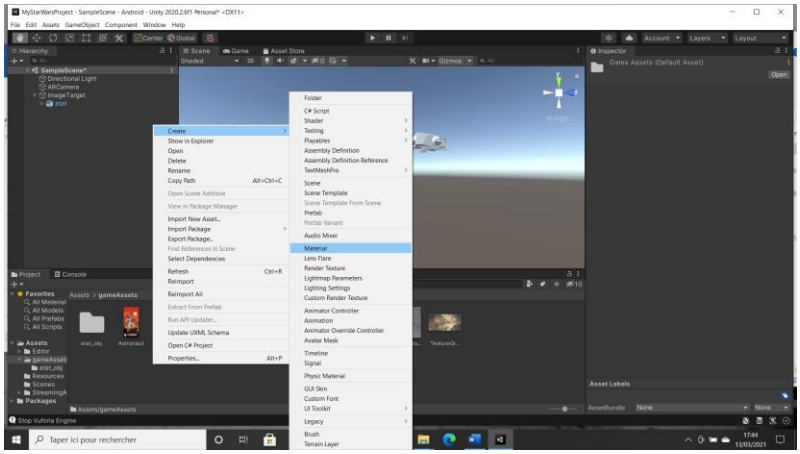
    - Ajouter ImageTarget et choisir une image :
        - de la dataBase Mars de Vuforia (astronaute, Oxygen…)
        - la vôtre : Il faut d'abord l'importer dans les Assets. Attention, votre image doit être assez riche en texture/information sinon la reconnaissance ne fonctionnera pas
    - Ajouter un Objet 3D, et faite le dériver de ImageTarget pour qu'il apparaisse sur l'image lorsque celle-ci est détectée par la caméra
    - Vérifiez la taille/position de l'objet par rapport à l'image, sinon vous risquez de ne pas voir l'objet car vous êtes à l'intérieur de celui-ci
    - Ajustez scale et position avec Unity3D pour la scène soit cohérente
    - 
- Lancer avec webcam (si image astronaute sur smartphone)



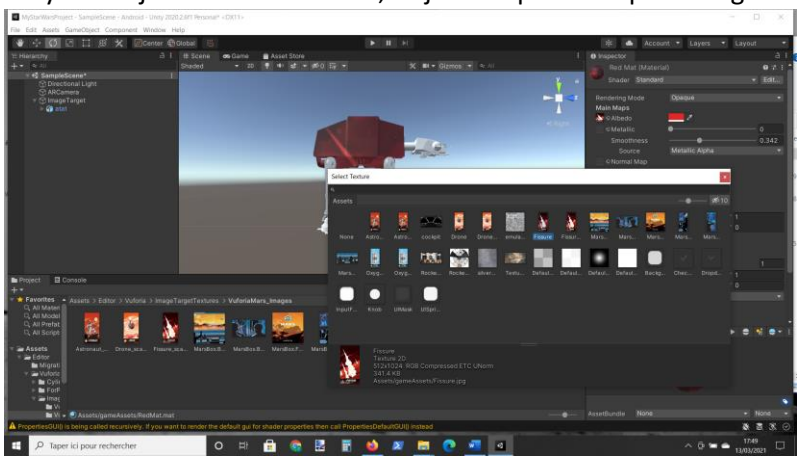
- Lancer ensuite avec android/smartphone (imprimer votre photo ou afficher sur PC pour tester)

Texturer un modèle 3D

- Récupérez le répertoire GameAssets sur mon Github (Modèle ATAT, cockpit, silver texture, baslter.wav)
- Drag&drop GameAsset dans Asset folder
- Vous pouvez remplacer l'objet 3D ajouté (sphère/box) par le modèle ATAT, le fichier .obj
- N'oubliez pas de vérifier l'échelle de l'objet et sa position par rapport l'image
- Quand le modèle est ajouté (objet fils de ImageTarget), vous pouvez lui ajouter une texture (silver ou autre)
- Pour créer une texture, il faut créer un matériel en cliquant droit dans le répertoire des Assets



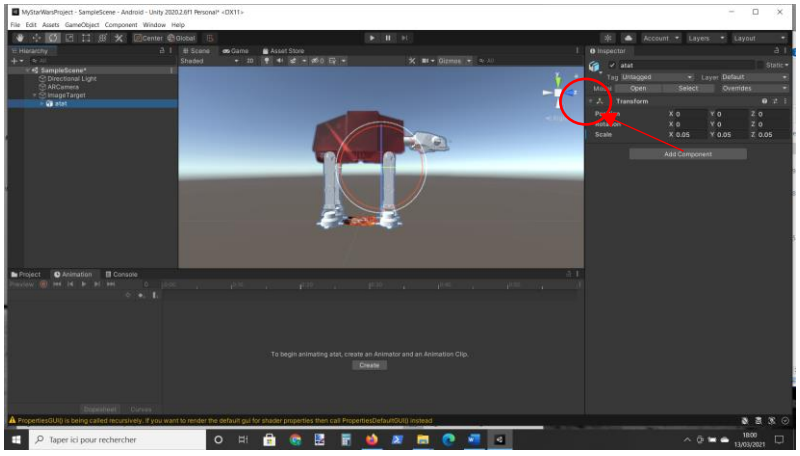
- Vous pouvez le renommer dans les prefabs, puis choisir sa couleur et/ou ajouter une texture en cliquant sur le symbole juste devant Albedo, ici j'utilise par exemple l'image Fissure et du rouge que l'on voit sur l'ATAT



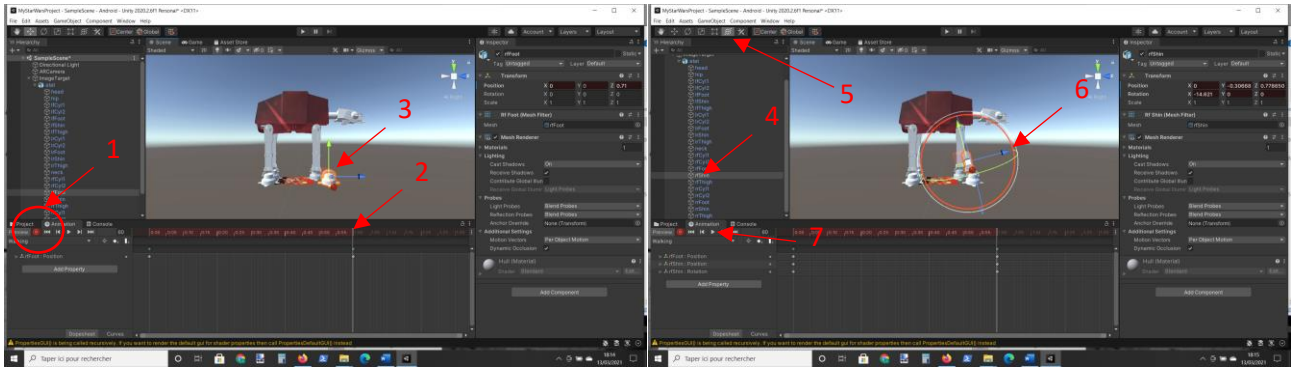
- Pour ajouter une texture à un objet il faut faire un glisser/déposer du matériel disponibles dans vos assets sur l'objet de votre scène.

Animation d'un modèle 3D

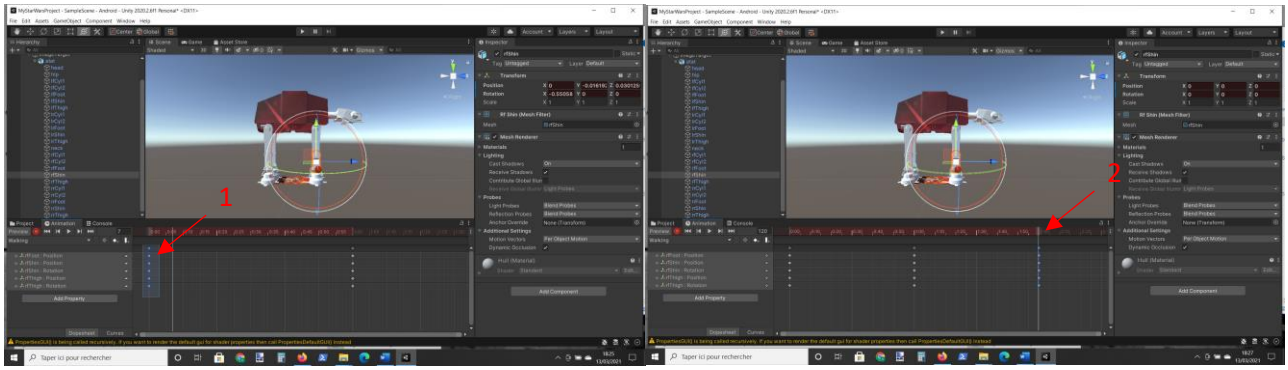
- Tout d'abord ajoutez la fenêtre Animation : Windows/Animation/Animation
- Déplacez cette fenêtre pour qu'elle devienne un onglet à côté de la console
- Recentrer la vue sur l'objet : double-clickez dessus. Cliquez ensuite sur l'axe de vue : X/Y/Z dans la fenêtre scene



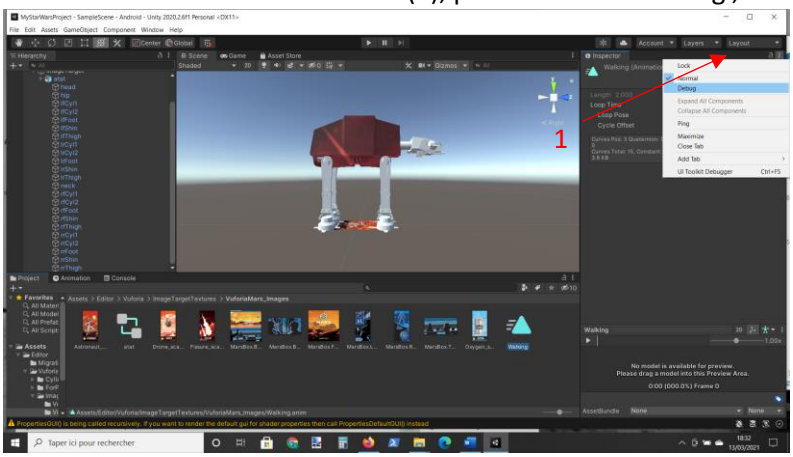
- Pour créer une animation de votre modèle, sélectionnez-le puis cliquez sur Create dans la fenêtre Animation et donnez un nom à celle-ci
- L'animation fonctionne par keyframe. Vous devez mettre votre modèle dans les différentes positions choisies à certains instants clefs comme pour un dessin animé. Unity va interpoler les mouvements entre ces positions. Dans la suite nous allons déplacer une jambe de l'ATAT pour simuler la marche.
- Toutes les modifications que vous allez effectuer doivent être effectuer dans le mode Recording. Il faut appuyer sur le bouton rouge avant (1), puis sélectionner un moment de l'animation (2) (1s ou 2s…) puis déplacer les différentes parties du modèle (3). S'il est difficile de sélectionner la partie dans la scène, sélectionnez là en cliquant dans la hiérarchie (4). Pour avoir des déplacements et rotation selon les axes, sélectionnez le mode Move/rotate/scale qui est le plus complet (5) puis déplacez vos parties (6). Cliquez enfin sur lecture (7) pour vérifier l'animation.



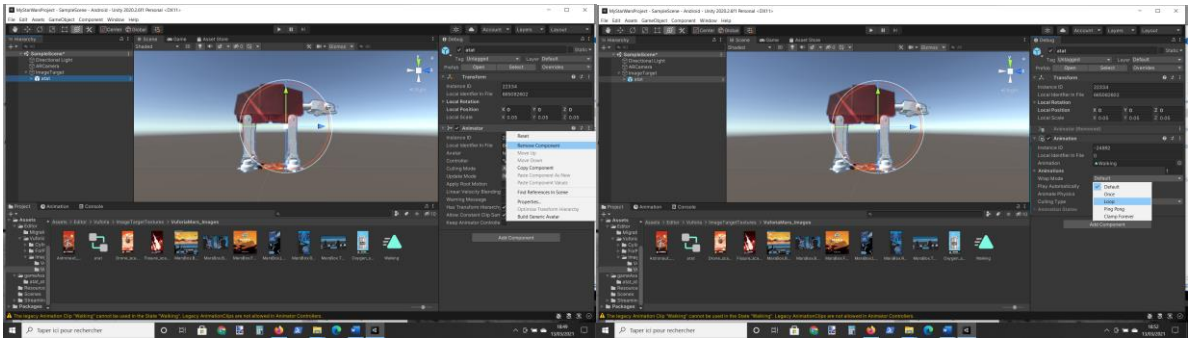
- Vous pouvez aussi copier/coller certaines positions de votre objet dans la fenêtre animation pour revenir à la position de départ par exemple. Sélectionnez (1), CTRL+C, cliquez au temps voulu (2), CTRL+V



- Une fois l'animation finie, faites CTRL+S pour la sauver. Celle-ci doit apparaitre dans vos Assets. Vous pouvez aussi générer un préfab de votre modèle ATAT animé en faisant un Drag&Drop du modèle dans la hierarchie vers les Assets.
- Normalement si vous testez avec votre WebCam ou votre smartphone, le modèle doit être animé lorsque votre image est détectée.
    - Si l'animation ne fonctionne pas.
        - Sélectionnez votre animation (1), passez en mode debug , sélectionner legacy

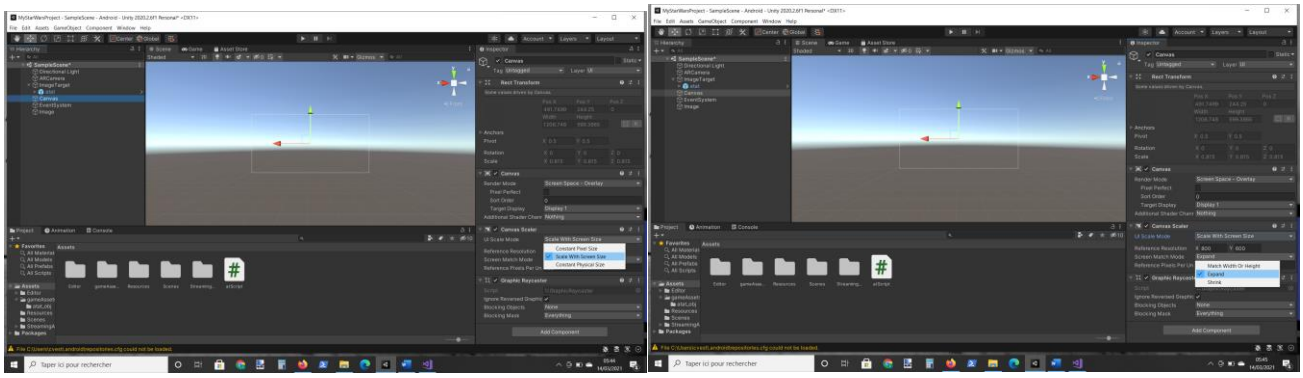


- Sélectionnez ensuite le modèle ATAT, supprimez la composante Animator et ajoutez une composante Animation. Faites un Drag&Drop de votre animation dans la case animation Clip. Sélectionnez ensuite le mode loop, ca devrait fonctionner.

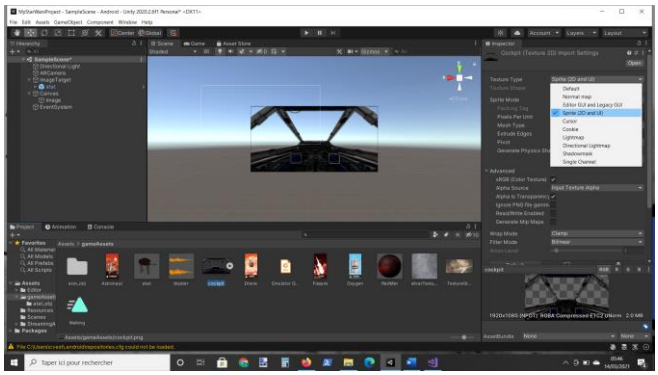


Simuler un cockpit ou ajouter un écran fixe

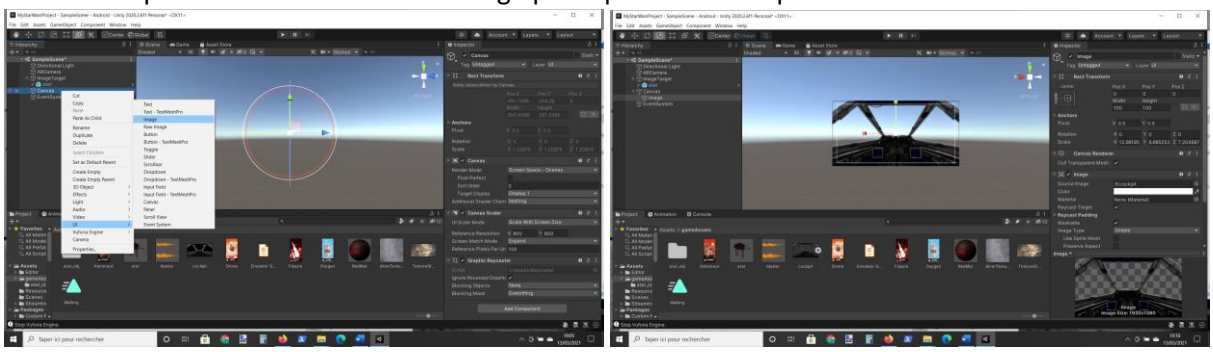
- Ajoutez à la scène un Canvas : UI/Canvas, celui-ci représente l'écran 2D de votre smartphone



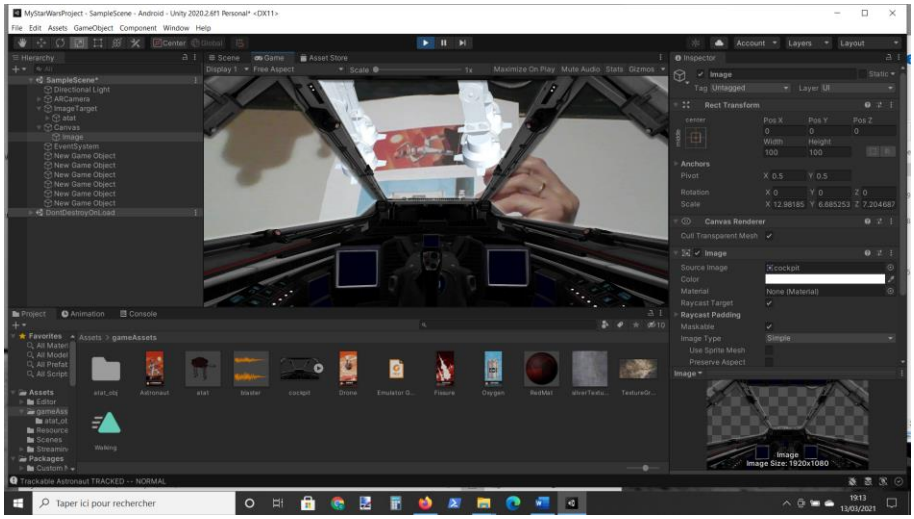
- Sélectionner votre image, le cockpit et transformez la en sprite 2D



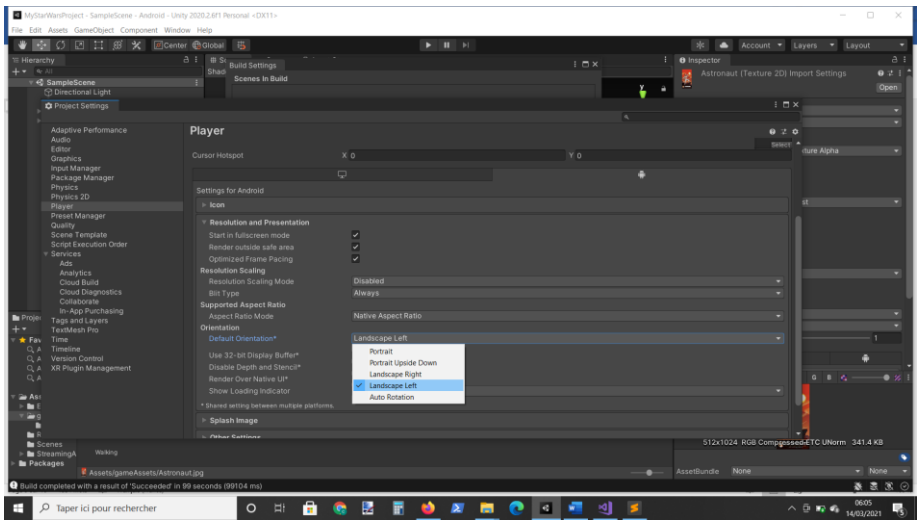
- Ajouter ensuite une image au Canvas, puis glissez-déposez votre imageSprite dans la case Source Image. Rescalez et positionnez ensuite votre image pour qu'elle soit adaptée au Canvas



- Si vous lancez l'application, le cockpit doit apparaitre maintenant



- Pour éviter que le display switch de mode portrait paysage, il faut choisir le comportement du smartphone pour les changements d'orientation dans Build Settings/Player Settings/ Orientation. Choisissez celui qui vous convient

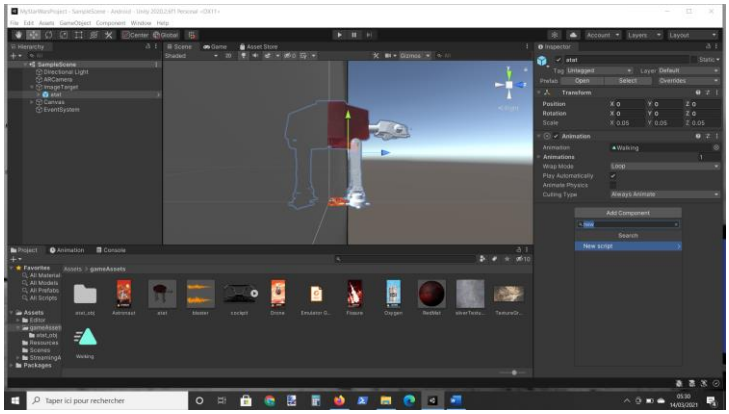


Amélioration avec des scripts C#

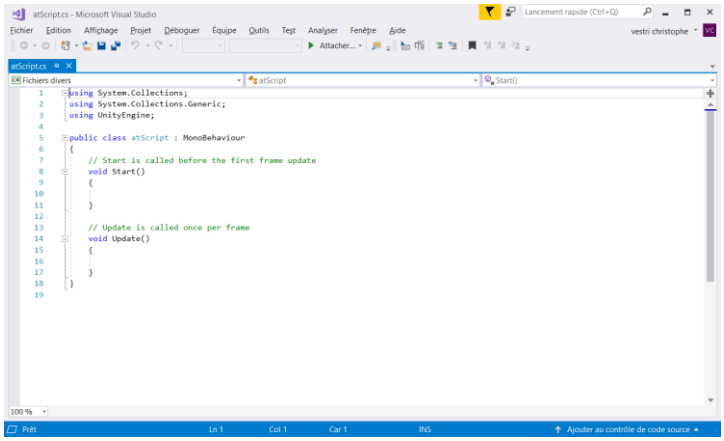
- Voici la doc pour comprendre le fonctionnement des scripts unity : https://docs.unity3d.com/Manual/ScriptingSection.html
- Pour débugger, il semble que ce soit Visual studio qu'il faille utiliser, même sous MacOS
- Les comportements des GameObjets sont contrôlés par les composants qui lui sont attachés. Unity permet de créer nos propres composants pour définir ces comportements pour déclencher des actions selon des évènements de l'application, modifier le comportement selon le temps ou répondre à une entrée utilisateur (click ou mouvement smartphone).
- Unity est basé sur Mono, toutes les classes dérivent de MonoBehaviour qui va permettre de définir les comportements en créant les fonctions nécessaires. Les principales sont init() et les suivantes https://docs.unity3d.com/Manual/EventFunctions.html
- Tous les variables publiques sont visibles dans l'inspector, c'est un des moyen de lier les objets entre eux
- Autre doc Unity : https://unity3d.com/fr/learning-c-sharp-in-unity-for-beginners

Déplacer un objet par script

- L'idée est de déplacer la position du GameObject tout au long du temps
- Sélectionnez le modèle que vous voulez déplacer (ici Atat), pour le déplacer il faut lui ajouter ce comportement qui sera décrit dans un script : cliquez « Add component » : New Script et nommez ce script atScript, puis cliquez Create and Add



- Maintenant , vous devez voir votre script atScript comme composante, double-cliquez dessus pour l'éditer

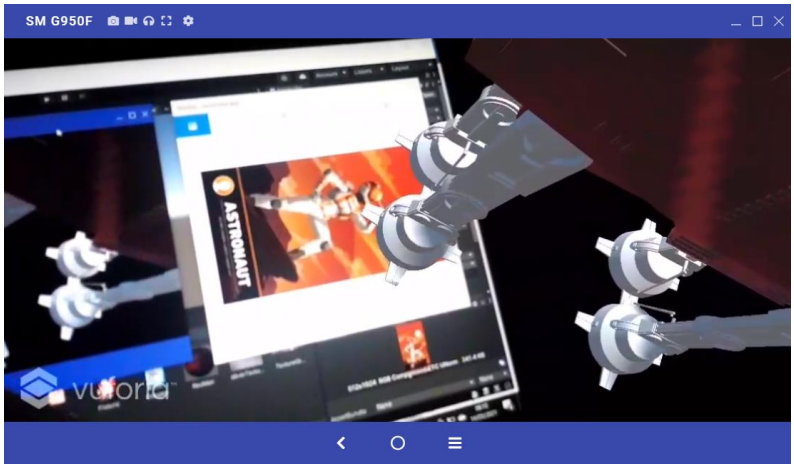


- Vous avez deux fonctions : Start() qui est appelée à la création du GameObject et Update() qui est appelée lors du calcul de chaque frame (environ 30fps selon). Voir https://docs.unity3d.com/Manual/ExecutionOrder.html
- Remplacez Update par la fonction suivante :

```
// Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * .01f * Time.deltaTime);
        transform.Rotate(Vector3.up * 3f * Time.deltaTime);
    }
```

- Le script l'objet d'une translation en avant et d'une rotation autour de la verticale

- Si vous relancez, le modèle doit se déplacer (Notez que j'ai enlevé le Canvas pour mieux voir le déplacement)

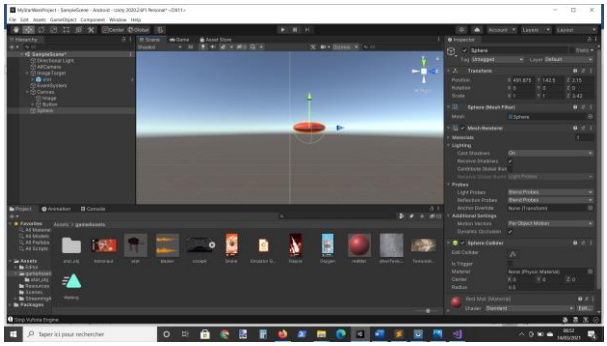


Ajout d'un bouton pour créer et déplacer un missile

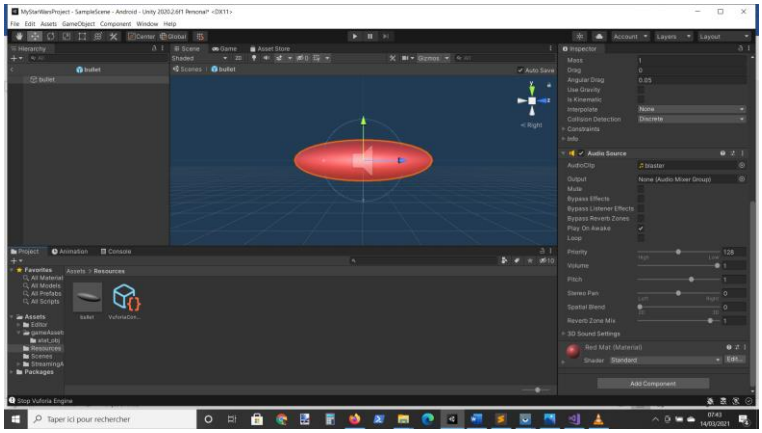
- Plusieurs fonctionnalités dans ce script. On va créer un bouton. Quand on cliquera dessus, celui -ci va créer un prefab (un missile : capsule rouge). Celui-ci va déclencher un son audio, se déplacer puis se détruire au bout de 3 secondes.
- Si vous avez supprimé le Canvas, remettez en un avec Cockpit (Simuler un cockpit ou ajouter un écran fixe)

Création du missile

- On va créer une capsule rouge que l'on va placer au centre du canvas, lui ajouter un son et la transformer en un prefab.
- Créez une sphère, double-cliquez pour zoomer dessus et clique sur Axe X pour avoir une caméra de côté par rapport au Canvas. Texturez-la avec un matériel rouge, donnez-lui une forme de capsule et placez-la devant le canvas.



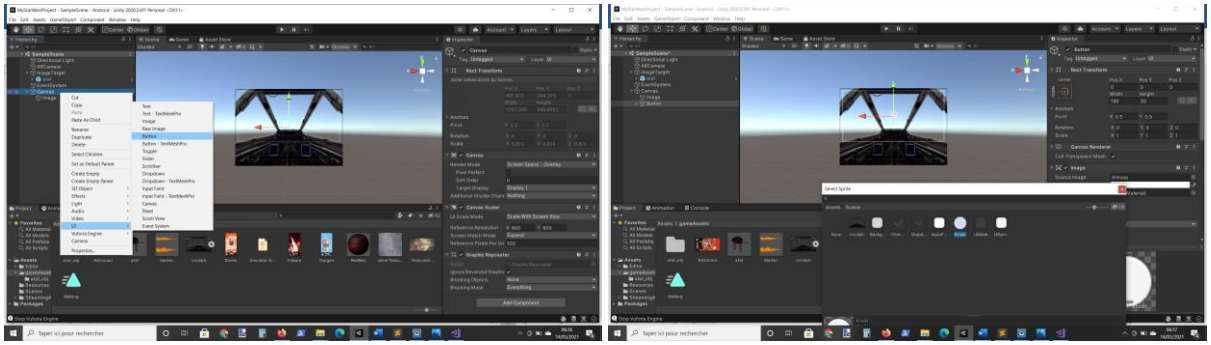
- Supprimez collider sphere component, ajoutez rigidBody component mais uncheckez gravity pour qu'elle puisse se déplacer en ligne droite.
- Renommez la Sphere « bullet » et créez un prefab en déplaçant bullet de la hiérarchie dans les Ressources. Ne vous trompez pas de nom ni de répertoire, c'est le nom du prefab créé par le script et Unity va chercher votre prefab dans Ressources. Ajoutez ensuite une composante audiosource et glissez-déposez le son Blaster.wav



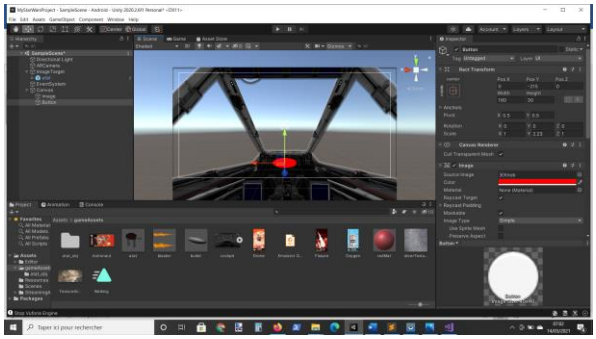
- Votre préfab est normalement prêt à fonctionner

Création du bouton

- Pour associer une fonction à un click de bouton il y a plusieurs manières de faire.
    - On peut ajouter un script avec la fonction à appeler à n'importe quel objet puis l'associer par glisser déposer avec la fonction Onclick() du bouton (ce qu'on va faire)
    - On peut aussi attacher des eventHandlers au bouton mais c'est plus complexe (https://docs.unity3d.com/2019.1/Documentation/ScriptReference/UI.Button-onClick.html) c'est ce qu'on voit souvent dans les tutos.
- On commence par ajouter un bouton et choisir comme texture un cercle (Knob),



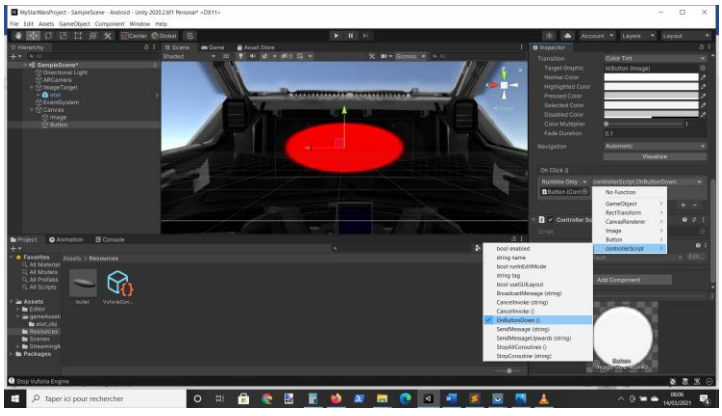
- Changez la couleur pour rouge, supprimez le texte et déplacez/rescalez le boutton pour le mettre au milieu du cockpit



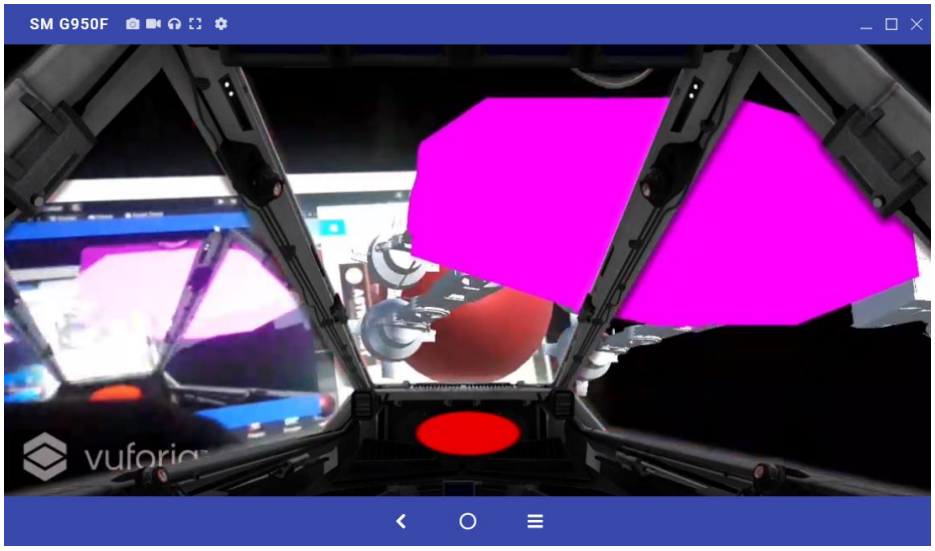
- Pour ajouter le comportement au click du bouton, il faut lui dire quelle fonction appeler. Pour cela on ajoute un nouveau script controllerScript au bouton (Add new Script). On édite le code et ajoute la fonction suivante :

```
// création, mvt et destruction du bullet
    public void OnButtonDown()
    {
        GameObject bullet = Instantiate(Resources.Load("bullet", typeof(GameObject))) as
 GameObject;
        Rigidbody rb = bullet.GetComponent<Rigidbody>();
        bullet.transform.rotation = Camera.main.transform.rotation;
        bullet.transform.position = Camera.main.transform.position;
        rb.AddForce(Camera.main.transform.forward * 500f);
        bullet.GetComponent<AudioSource>().Play();
        Destroy(bullet, 3);
    }
```

- Il faut ensuite attacher cette fonction à la fonction onclick() du boutton par glissé/déposé et recherche de la fonction dans le menu.



- Normalement tout fonctionne et vous pouvez tirer et entendre le son



Autres fonctionnalités ou tests

Roll a ball

- Rool a ball est un petit jeu qui permet d'apprendre d'autres fonctionnalités :
  - gérer les input pour contrôler le mouvement un objet,
  - attacher la caméra à cet objet pour le suivre
  - détecter les collisions et y associer une action
  - Gérer et afficher un score
- Online tutorial : https://learn.unity.com/project/roll-a-ball
- Ce qui est intéressant, c'est non pas d'utiliser un keyboard mais l'orientation du smartphone, pour cela, il suffit récupérer l'accélération. Voici un script permettant de gérer le clavier et l'accélération avec smartphone (à vérifier qu'il fonctionne toujours, sinon dites-moi svp)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerControl : MonoBehaviour
{
    private Rigidbody rb;

    public float speed;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        if (SystemInfo.deviceType == DeviceType.Desktop)
        {
            float moveHorizontal = Input.GetAxis("Horizontal");
            float moveVertical = Input.GetAxis("Vertical");
            Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
            rb.AddForce(movement * speed);
        }
        else
        {
            float moveHorizontal = Input.acceleration.x;
            float moveVertical = Input.acceleration.y;
            Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
            rb.AddForce(movement * speed);
        }
```

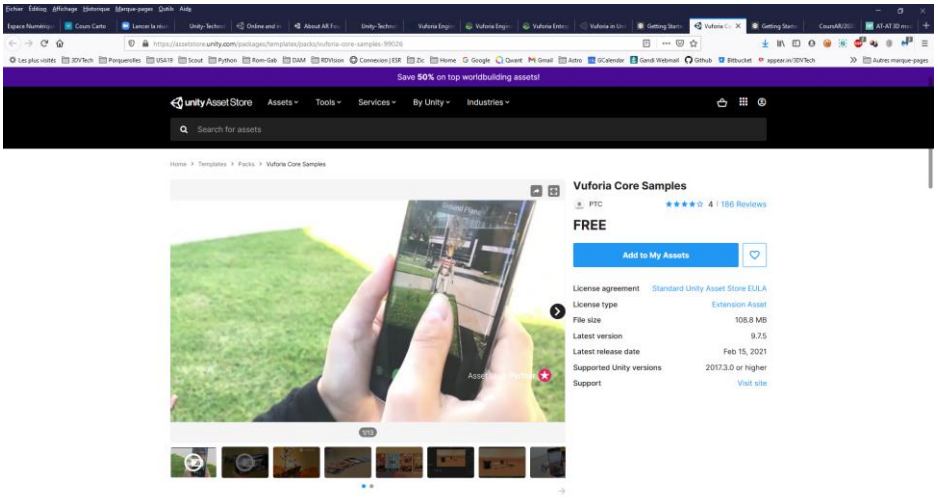
```
        }
    }
```

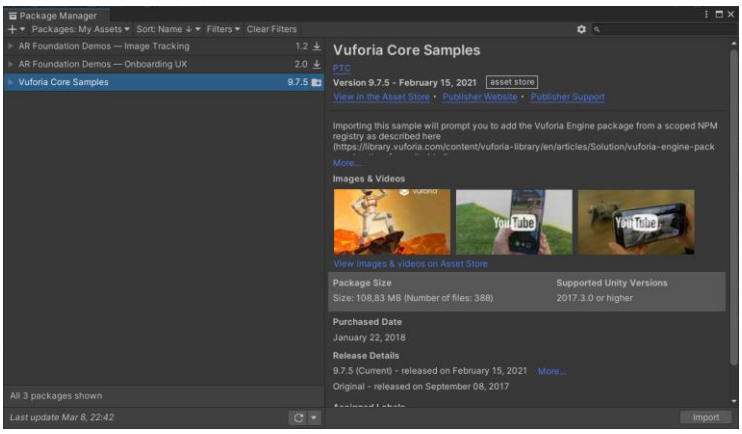
Géolocalisation

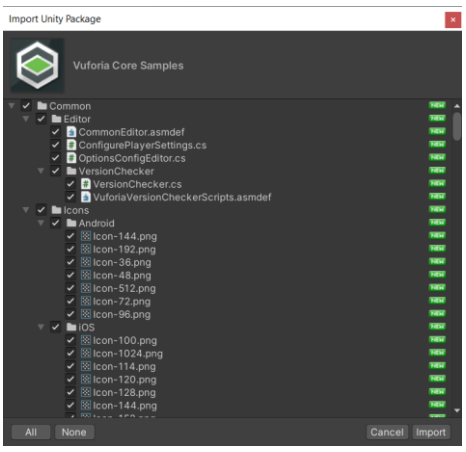
- La géolocalisation est clef de la réalité augmentée. Il existe plusieurs manières de les gérer mas ce n'est pas simple. En voici quelques-unes :
    - Un package qui permet de créer des objets géolocalisés
      https://noperation.wordpress.com/2020/05/24/lar/
    - Avec MapBox pour gérer et se déplacer dans des cartes Géolocalisées :
      https://docs.mapbox.com/unity/maps/guides/
      Un tuto avec différents essais qui explique pourquoi utilise MapBox
      https://www.youtube.com/watch?v=_Idze_NzDu0
    - Avec Unity il semble qu'il n'y ai pas encore d'outils pour ca, car trop complexe:
        - On peut toutefois connaitre sa localisation, pour cela il faut lancer le service :
          https://docs.unity3d.com/ScriptReference/LocationService.Start.html
        - Pour générer des GameObjets géolocalisés, c'est plus compliqué, ici un petit tuto mais pas simple, si vous trouvez mieux, dites le moi : https://www.instructables.com/How-to-Markerless-GPS-Based-Augmented-Reality/

Tester les Vuforia core samples :

- Vous pouvez tester les Ground Plane Detection et autres technos de Vuforia avec les samples
- Aller dans l'assetStore et ajouter VuforiaCoreSamples dans votre liste d'assets
- https://assetstore.unity.com/packages/templates/packs/vuforia-core-samples-99026
- Ensuite : package manager/my assets/vuforia core sample import
- Il ne vous reste plus qu'a compiler et tester, voici les technos Vuforia :
  https://library.vuforia.com/features/overview.html







Particules :

- Pour ajouter un système de particules (feu, brouillard, pluie…) il vous faut rajouter un effect

- Click droit, Effect/particleSystems
- Ensuite vous pouvez régler tout un tas de propriétés : https://learn.unity.com/tutorial/introduction-to-particle-systems